

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**
**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №2.1

Дисциплина: «Основы кроссплатформенного программирования»

Тема: «Основы языка Python»

Выполнил: студент 1 курса

группы ИВТ-б-о-21-1

Горшков Виталий Игоревич

Ставрополь 2022

Выполнение работы:

1. Создал репозиторий GitHub с лицензией MIT, добавил .gitignore с ЯП python, клонировал репозиторий на ПК и организовал репозиторий согласно модели ветвления git-flow:

The screenshot displays a GitHub repository interface. At the top, there are navigation elements: a dropdown menu for the 'main' branch, a link to '1 branch', and a link to '0 tags'. To the right are buttons for 'Go to file', 'Add file', and a green 'Code' button. Below this is a table of commit history. The first commit is by 'VitaliyPitsa' titled 'Update README.md', with commit hash 'dfb7358', made '10 minutes ago', and containing '2 commits'. Below the commit history is a section for the 'README.md' file. It shows the file name with an edit icon, followed by the title 'lab2.1' in a large font. Below the title is a horizontal line and the text 'Горшков Виталий ИВТ-б-о-21-1'.

Commit Hash	Author	Message	Time
dfb7358	VitaliyPitsa	Update README.md	10 minutes ago
		Initial commit	11 minutes ago
		Initial commit	11 minutes ago
		Update README.md	10 minutes ago

README.md

lab2.1

Горшков Виталий ИВТ-б-о-21-1

Рисунок 1 - Создание репозитория

```

Microsoft Windows [Version 10.0.19043.1645]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Виталий>cd /d C:\lab2.1

C:\lab2.1>git clone https://github.com/VitaliyPitsa/lab2.1.git
Cloning into 'lab2.1'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.

C:\lab2.1>ss

```

Рисунок 2 - Клонирование репозитория

```

C:\lab2.1>git flow init
Initialized empty Git repository in C:/lab2.1/.git/
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/lab2.1/.git/hooks]

```

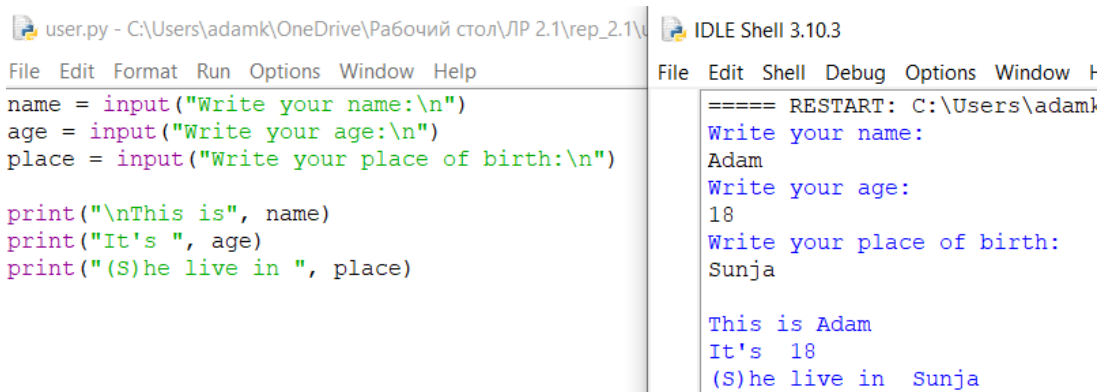
Рисунок 3 - Организация репозитория согласно модели ветвления git-flow

2. Написал программу user.py, которая запрашивала бы у пользователя имя, возраст и место жительства, после этого выводила бы 3 строки

```

"This is `имя`"
"It is `возраст`"
"(S)he live in `место_жительства`"

```



The screenshot shows a Python script named 'user.py' in a text editor and its execution in the IDLE Shell. The script prompts the user for their name, age, and place of birth, then prints the results in a specific format.

```

user.py - C:\Users\adamk\OneDrive\Рабочий стол\ЛР 2.1\rep_2.1\
File Edit Format Run Options Window Help
name = input("Write your name:\n")
age = input("Write your age:\n")
place = input("Write your place of birth:\n")

print("\nThis is", name)
print("It's ", age)
print("(S)he live in ", place)

```

The IDLE Shell output shows the execution of the script with the following input and output:

```

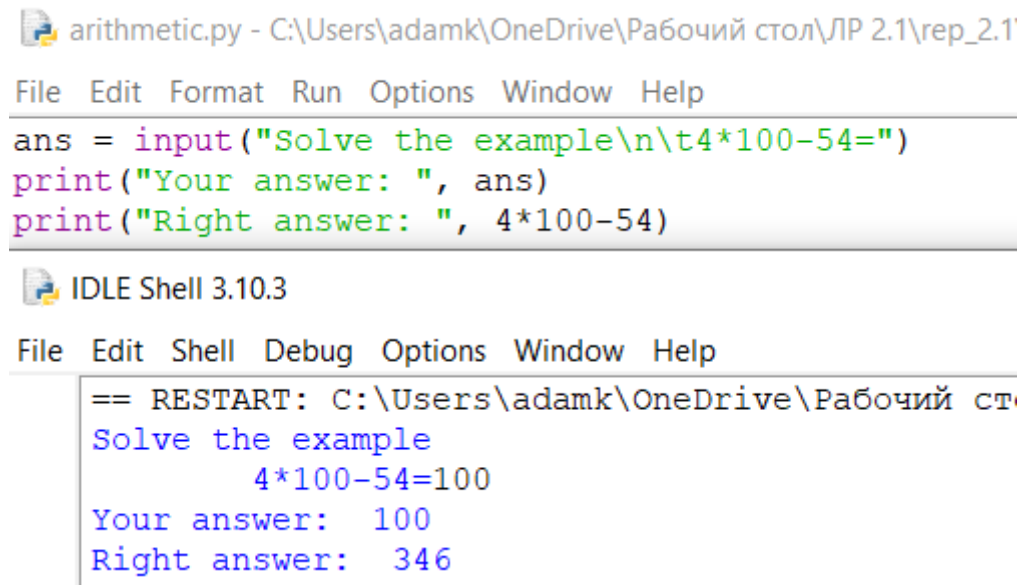
==== RESTART: C:\Users\adamk\
Write your name:
Adam
Write your age:
18
Write your place of birth:
Sunja

This is Adam
It's 18
(S)he live in Sunja

```

Рисунок 4 - Программа user в репозитории

4. . Написал программу (файл arithmetic.py), которая предлагала бы пользователю решить пример $4 * 100 - 54$. Потом выводила бы на экран правильный ответ и ответ пользователя.



```
arithmetic.py - C:\Users\adamk\OneDrive\Рабочий стол\ЛР 2.1\rep_2.1'
```

```
File Edit Format Run Options Window Help
```

```
ans = input("Solve the example\n\t4*100-54=")
print("Your answer: ", ans)
print("Right answer: ", 4*100-54)
```

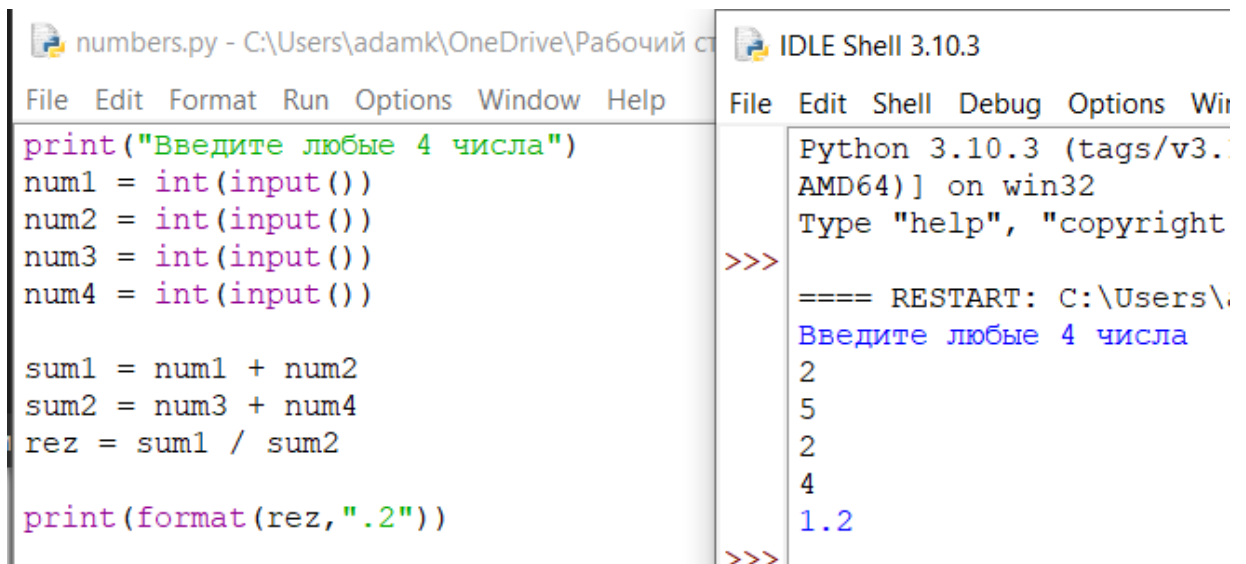
```
IDLE Shell 3.10.3
```

```
File Edit Shell Debug Options Window Help
```

```
== RESTART: C:\Users\adamk\OneDrive\Рабочий ст
Solve the example
      4*100-54=100
Your answer:  100
Right answer: 346
```

Рисунок 5 - Программа arithmetic.py

5. Написал программу numbers.py, которая запрашивает у пользователя 4 числа, отдельно складывает первые два и вторые два, затем делит первую сумму на вторую, после выводит рез-т на экран с точностью до сотен.



```
numbers.py - C:\Users\adamk\OneDrive\Рабочий стол
File Edit Format Run Options Window Help
print("Введите любые 4 числа")
num1 = int(input())
num2 = int(input())
num3 = int(input())
num4 = int(input())

sum1 = num1 + num2
sum2 = num3 + num4
rez = sum1 / sum2

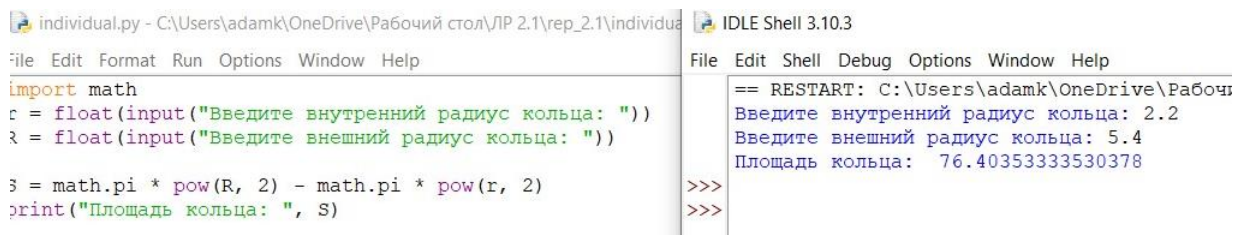
print(format(rez, ".2"))

IDLE Shell 3.10.3
File Edit Shell Debug Options Window Help
Python 3.10.3 (tags/v3.10.3:amd64) on win32
Type "help", "copyright", and "credits()" for more
>>>
==== RESTART: C:\Users\adamk\OneDrive\Рабочий стол
Введите любые 4 числа
2
5
2
4
1.2
>>>
```

Рисунок 6 - Программа numbers.py

5. Написал программу для индивидуального задания:

Найти площадь кольца по заданным внешнему и внутреннему радиусам.



```
individual.py - C:\Users\adamk\OneDrive\Рабочий стол\ЛР 2.1\rep_2.1\individual.py
File Edit Format Run Options Window Help
import math
r = float(input("Введите внутренний радиус кольца: "))
R = float(input("Введите внешний радиус кольца: "))

S = math.pi * pow(R, 2) - math.pi * pow(r, 2)
print("Площадь кольца: ", S)

IDLE Shell 3.10.3
File Edit Shell Debug Options Window Help
== RESTART: C:\Users\adamk\OneDrive\Рабочий стол
Введите внутренний радиус кольца: 2.2
Введите внешний радиус кольца: 5.4
Площадь кольца: 76.4035333530378
>>>
>>>
```

Рисунок 7 - Программа индивидуального задания

6. Сделал коммит изменений в ветку разработки, выполнил ее слияние с веткой main и отправил сделанные изменения на уд. репозиторий.

```
C:\lab2.1\lab2.1>git branch
* develop
  main

C:\lab2.1\lab2.1>git add .

C:\lab2.1\lab2.1>git commit -m "added programms"
[develop 353763e] added programms
 5 files changed, 41 insertions(+)
 create mode 100644 8povsl.py
 create mode 100644 arithmetic.py
 create mode 100644 individual.py
 create mode 100644 numbers.py
 create mode 100644 user.py
```

Рисунок 8 - Коммит изменений в ветку develop

```
C:\lab2.1\lab2.1>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\lab2.1\lab2.1>git merge develop
Updating dfb7358..353763e
Fast-forward
 8povsl.py      | 7 ++++++
 arithmetic.py  | 7 ++++++
 individual.py  | 8 ++++++
 numbers.py     | 8 ++++++
 user.py        |11 ++++++++
 5 files changed, 41 insertions(+)
 create mode 100644 8povsl.py
 create mode 100644 arithmetic.py
 create mode 100644 individual.py
 create mode 100644 numbers.py
 create mode 100644 user.py
```

Рисунок 9 - Слияние ветки develop с веткой main

```
C:\lab2.1\lab2.1>git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 858 bytes | 858.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/VitaliyPitsa/lab2.1.git
 dfb7358..353763e  main -> main
```

Рисунок 10 - push коммитов на уд. репозиторий

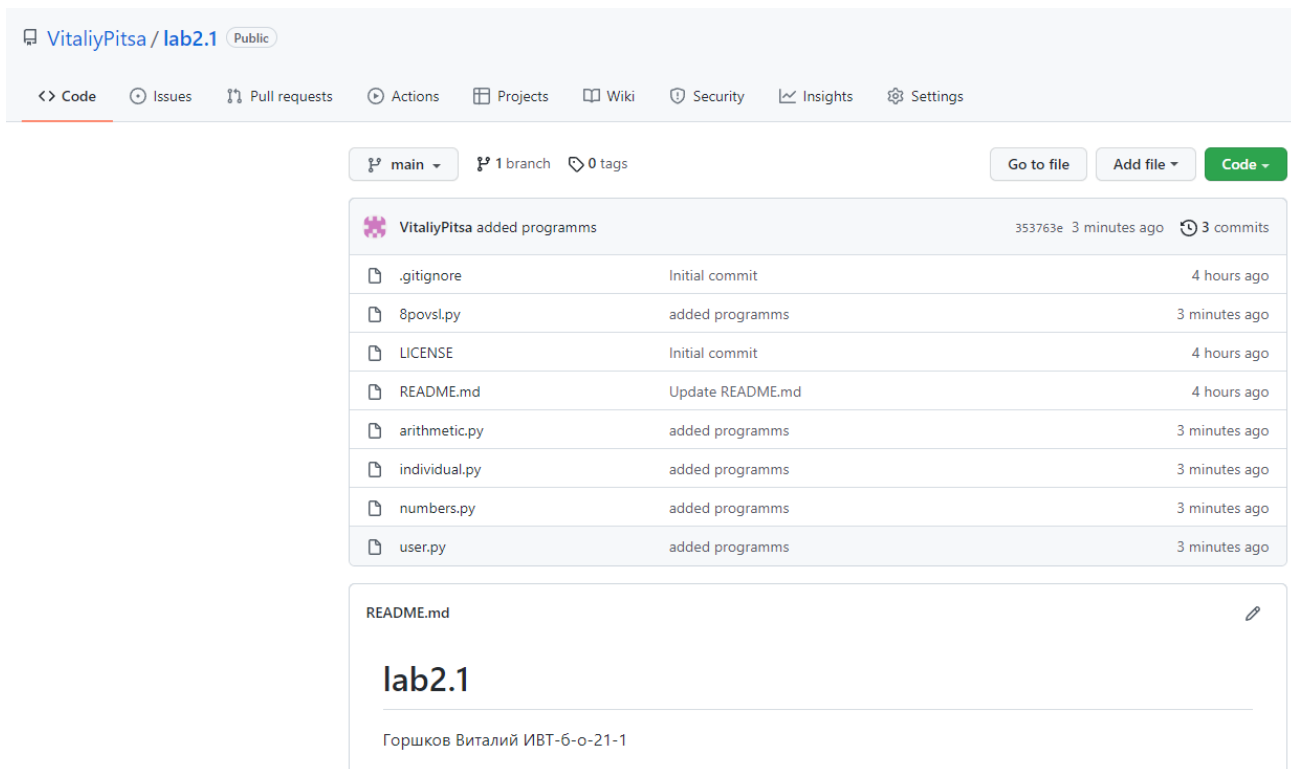


Рисунок 11 - Изменения на уд. сервере

1. Опишите основные этапы установки Python в Windows и Linux.

Linux: Чаще всего интерпретатор Python уже входит в состав дистрибутива.

Windows: Оsn. этапы установки Python на Windows:

- 1) Скачать дистрибутив с официального сайта;
- 2) Запустить скачанный установочный файл;
- 3) Выбрать способ установки;
- 4) Отметить необходимые опции установки;
- 5) Выбрать место устновки;
- 6) Готово.

2. В чем отличие пакета Anaconda от пакета Python, скачиваемого с официального сайта?

Пакет Anaconda содержит версии языка Python 2 и 3, набор наиболее часто используемых библиотек и удобную среду разработки и исполнения, запускаемую в браузере, а также на Anaconda удобнее запускать примеры.

3. Как осуществить проверку работоспособности пакета Anaconda?

Для выполнения проверки работоспособности Anaconda необходимо вначале запустить командный процессор с поддержкой виртуальных окружений Anaconda. В появившейся командной строке необходимо ввести `> jupyter notebook`, в результате чего отобразится процесс загрузки веб-среды Jupyter Notebook, после чего запустится веб-сервер и среда разработки в браузере. Создать ноутбук для разработки, для этого нажать на кнопку New и в появившемся списке выбрать Python. В результате будет создана новая страница в браузере с ноутбуком. Ввести в первой ячейке команду `print("Hello, World!")` и нажать Alt+Enter на компьютере. Ниже ячейки должна появиться соответствующая надпись.

4. Как задать используемый интерпретатор языка Python в IDE PyCharm?

Указать путь до интерпретатора в настройках IDE, для этого:

- 1) Нажмите на шестеренку в верхнем правом углу, выберите "Add..".
- 2) Далее выберите "System Interpreter";
- 3) Нажмите на 3 точки "..." справа от поля в выборе интерпретатора;
- 4) Укажите путь до интерпретатора.

5. Как осуществить запуск программы с помощью IDE PyCharm?

Сочетанием клавиш Shift+F10.

6. В чем суть интерактивного и пакетного режимов работы Python?

Интерактивный.

Python можно использовать как калькулятор для различных вычислений, а если дополнительно подключить необходимые математические библиотеки, то по своим возможностям он становится практически равным таким пакетам как Matlab, Octave и т.п.

Проектный.

В этом режиме сначала записывается вся программа, а потом эта программа выполняется полностью.

7. Почему язык программирования Python называется языком динамической типизации?

Т. к. в ЯП Python проверка типа происходит во время выполнения, а не компиляции.

8. Какие существуют основные типы в языке программирования Python?

Типы в ЯП Python:

1. None
2. Логические переменные
3. Числа
4. Списки
5. Строки
6. Бинарные списки
7. Множества
8. Словари

9. Как создаются объекты в памяти? Каково их устройство? В чем заключается процесс объявления новых переменных и работа операции присваивания?

Для того, чтобы объявить и сразу инициализировать переменную необходимо написать её имя, потом поставить знак равенства и значение, с которым эта переменная будет создана.

При инициализации переменной, на уровне интерпретатора, создается целочисленный объект, который имеет некоторый идентификатор, значение и тип. Посредством оператора "=" создается ссылка между переменной и объектом.

10. Как получить список ключевых слов в Python?

Список ключевых слов можно получить непосредственно в программе, для этого нужно подключить модуль keyword и воспользоваться командой keyword.kwlist.

11. Каково назначение функций id() и type()?

Функция id() предназначена для получения значения идентичности объекта.

С помощью функции type() можно получить тип конкретного объекта.

12. Что такое изменяемые и неизменяемые типы в Python.

К неизменяемым (immutable) типам относятся: целые числа (int), числа с плавающей точкой (float), комплексные числа (complex), логические переменные (bool), кортежи (tuple), строки (str) и неизменяемые множества (frozenset).

К изменяемым (mutable) типам относятся: списки (list), множества (set), словари (dict).

13. Чем отличаются операции деления и целочисленного деления?

При целочисленном делении отбрасывается дробная часть от деления чисел, при операции деления дробная часть не отбрасывается.

14. Какие имеются средства в языке Python для работы с комплексными числами?

Для создания комплексного числа можно использовать функцию complex(a, b), в которую, в

качестве первого аргумента, передается действительная часть, в качестве второго – мнимая.

Либо записать число в виде $a + bj$. Комплексные числа можно складывать, вычитать, умножать, делить и возводить в степень. У комплексного числа можно извлечь действительную (`x.real`) и мнимую части (`x.imag`).

Для получения комплексносопряженного число необходимо использовать метод `conjugate()`.

15. Каково назначение и основные функции библиотеки (модуля) `math`? По аналогии с модулем `math` изучите самостоятельно назначение и основные функции модуля `cmath`.

Для выполнения математических операций необходим модуль `math`.

Осн. операции библиотеки `math`:

`math.ceil(x)` - возвращает ближайшее целое число большее, чем `x`.

`math.fabs(x)` - возвращает абсолютное значение числа.

`math.factorial(x)` - вычисляет факториал `x`.

`math.floor(x)` - возвращает ближайшее целое число меньшее, чем `x`.

`math.exp(x)` - вычисляет $e^{**}x$.

`math.log2(x)` - логарифм по основанию 2.

`math.log10(x)` - логарифм по основанию 10.

`math.log(x[, base])` - по умолчанию вычисляет логарифм по основанию `e`, дополнительно можно указать основание логарифма.

`math.pow(x, y)` - вычисляет значение `x` в степени `y`.

`math.sqrt(x)` - корень квадратный от `x`.

`math.cos(x)` - косинус от `x`.

`math.sin(x)` - синус от `x`.

`math.tan(x)` - тангенс от `x`.

`math.acos(x)` - арккосинус от `x`.

`math.asin(x)` - арксинус от `x`.

`math.atan(x)` - арктангенс от `x`.

`math.pi` - число пи.

`math.e` - число `e`.

16. Каково назначение именных параметров `sep` и `end` в функции `print()`?

Через параметр `sep` можно указать отличный от пробела разделитель строк.

Параметр `end` позволяет указывать, что делать, после вывода строки.

17. Каково назначение метода `format()`? Какие еще существуют средства для форматирования строк в Python? Примечание: в дополнение к рассмотренным средствам изучите самостоятельно работу с f-строками в Python.

Форматирование может выполняться в так называемом старом стиле или с помощью строкового метода `format`.

Символы `%s` , `%d` , `%f` подставляются значения переменных. Буквы `s`, `d`, `f` обозначают типы данных – строку, целое число, вещественное число.

18. Каким образом осуществить ввод с консоли значения целочисленной и вещественной переменной в языке Python?

Указать перед `input` тип данных: `int(input())`.